# A unified formalism for side-channel and fault attacks on cryptographic circuits

Bruno Robisson, Hélène Le Bouder, Jean-Max Dutertre, Assia Tria

# A unified formalism for side-channel and fault attacks on cryptographic circuits

Bruno Robisson*, Hélène Le Bouder †, Jean-Max Dutertre† and Assia Tria*
*CEA-LETI †École Nationale Supérieure des Mines de Saint-Étienne
*†Département Systèmes et Architectures Sécurisés, Gardanne, France
{bruno.robisson, assia.tria}@cea.fr {lebouder, dutertre}@emse.fr

***Abstract*—Security is a key component for information technologies and communication. Security is a very large research area involved in the whole information technology, related to both hardware and software. This paper focuses on hardware security, and more specifically on hardware cryptanalysis whose aim is to extract confidential information (such as encryption keys) from cryptographic circuits. Many physical cryptanalysis techniques have been proposed in the last ten years but they always belong to one of those very distinct categories: fault and side channel attacks. In this article, a formal link between these two categories is proposed. To the best of our knowledge, this is the first time that a wide class of attacks is described in such a generic manner.**

**Keywords.** Differential Power Analysis, Differential Fault Analysis, Differential Behavioral Analysis, Template Attacks, Fault Sensitivity Analysis, $AES_{128}$.

## INTRODUCTION

Security is a key component for information technologies and communication. Among the security threats, a very important one is certainly due to vulnerabilities of the integrated circuits that implement cryptographic algorithms to ensure confidentiality, authentication or data integrity (such as smartcards). With the access to one of these circuits, the attacker tries either to reconstruct the functionality of the circuit (*reverse engineering*) or to recover cryptographic materials when the cryptographic algorithm is known (*physical or hardware cryptanalysis*). Both threats share a set of techniques. The first one, called *side channel attacks*, consists in observing some physical characteristics which are modified during the circuit's computation. The second technique, called *fault attacks*, consists in disrupting the circuit's behavior. The third one consists in getting information about the chip design by direct inspection of its structure. This inspection may be performed by using any kind of imaging techniques or by using destructive means such as abrasion, chemical etching or focused ion beam. Many protections have been proposed to improve security against those types of attacks. But every protection against physical cryptanalysis usually lacks of genericity and is usually very specific to one precise type of attack.

Even if physical cryptanalysis methods seem very different at first sight, several works has been proposed to described them with a commun formalism [MR03], [MOS09]. But these works only cover side channel attacks. This paper describes a formalism which is commun to both fault and side channel attacks. The short-term aim of this work is to define "new" attacks as new combinations of these concepts and algorithms. Then, we plan to provide efficient and modular implementation of these attacks. The long-term aim of this work is to merge the advantages of attack-specific protections to enable a more generic set of countermeasures.

In the first part of this article, we shortly describe side channel and fault attacks and give an informal explanation of the principles which are common in these two types of attack. The cryptographic algorithm used as example, $AES_{128}$ is also described. In the second part of the paper, we present a small set of definitions and the algorithms we use to extract information from physical attacks. Finally, we show that our proposed formalism easily fits with several representative examples.

## I. SCOPE OF THE WORK

### A. Physical cryptography

This paper deals with physical cryptanalysis of both software or hardware implementation of cryptographic algorithms (such as $AES_{128}$ described in I-C). These algorithms encrypt a plaintext with a key to obtain a ciphertext. Physical cryptanalysis is mainly based on the two following techniques:

*1) Side channel:* Side channel analysis exploits the fact that some physical values or "side channels" (such as power consumption, electromagnetic radiation or computation time) of a circuit depend on the key value [Koc96], [KJJ99], [SMP07]. A more rough technique consists in measuring directly some key-related internal computations by using micro-probing techniques. There are two subcategories of key recovering techniques based on side channel measurements. The first one, called DPA-like, consists in building a set of mathematical models (i.e. mathematical formulæ) from *a priori* knowledge about the circuit. Each model is associated with an hypothesis on the value of the key. Then, the models are compared with measurements. The model which matches the best with measurements is generally associated with the right key hypothesis. The second kind of side channel attacks needs a profiling step on another circuit. This circuit is supposed to be identical to the target and the attacker is supposed to be able to set the key value. In this case, the profiling step is used either to improve the model *a priori* (*stochastic attacks*) or to build a statistical model only based on measurements (*template attacks*).

*2) Faults:* Fault attacks consist in disrupting the circuit's behavior. Their aim is to alter the correct progress of the algorithm. Faults are injected into the device using various means such as laser, clock glitches, spikes on the voltage supply or electromagnetic perturbations. A more rough technique consists in modifying the circuit's operation by modifying internal computation through micro-probes, or even modifying the circuit itself by using focused ion beam. There are three subcategories of key recovering techniques that use the results of faults attacks. *Algorithm modifications* consist either in reducing the ciphering complexity of the cryptographic algorithm [CT05] or in bypassing hardware or software protections. *Differential Fault Attack* (DFA), originally described in [BDL97], [BS97] and in enhanced in [PQ03], [MSS06], [Muk09], [TM09], [Gir05], [RLK11], consists in retrieving the key by comparing the correct ciphertexts with faulty ones. A detailed comparison of DFA schemes against AES, for example, is given in [SLIO12]. To perform the third kind of fault attacks, called *safe-error attacks* [YJ00], [RM07], [LSG+10], the attacker does not necessarily need pairs of correct and faulty ciphertexts but only some information about the chip's behavior.

### B. Principle of the considered class of physical attacks

This paper does not deal yet with attacks which aim to modify the algorithms (by using fault or circuit modifications). It focuses on the different kinds of side channel attacks (DPA-like, stochastic or template), DFA and safe-error attacks.

For such attacks, physical access to the circuit gives information about the internal computations of the cryptographic algorithm. Thanks to these pieces of information, the attacker is able to apply a divide and conquer approach. Instead of retrieving the whole key in one step (typically constituted of one hundred to one thousand bits), the attacker performs the same elementary attack step several times. Every elementary attack enables him to retrieve a small part of the key, called a partial key (typically 8 bits for $AES_{128}$). At the end, the attacker is able to rebuild the whole key. We claim that the elementary steps of the attacks share the following method:

- Launch cryptographic operations on the target. Each launch is called *an experiment*. Different kinds of experimental data such as power, electromagnetic emission (EM), inputs or outputs, detailed in II-A, are recorded during these experiments.
- Make sure that some of these measurements are related to some others through the partial key. A more formal definition of "relations" between measurements and the key is provided in II-B.
- Build models that explain the relationship between these measurements. These models are parametrized with one value of partial key. These models are mathematical formulæ determined either by a priori knowledge on the chip or by using additional measurements. Examples of models used for attacks are described precisely in II-C.
- Compare the models to the measurements, by using the algorithms described in II-D. The best-fitted models with the measurements are generally models associated with the partial key.

To illustrate our formalism, we use as main example the AES crypto-algorithm shortly described above.

### C. Example of cryptographic algorithm : $AES_{128}$

AES is a standard established by the NIST [NIS01] for symmetric key cryptography. In this paper, we focus on the 128-bit-key version of the AES (represented in Figure 1). This algorithm ciphers a 128-bit long data (called *plaintext* or *input*) by using a 128-bit long key to obtain a 128-bit long data (called *ciphertext* or *output*). The encryption consists first in transforming the input data into a two-dimensional array of bytes, called the *state*. Then, after a preliminary XOR between the input and the key, $AES_{128}$ executes 10 times a function (called *round*) that operates on the state. The operations used during these rounds are the following ones:

- SubBytes is a non-linear transform working independently on individual bytes of the State. The result of this operation at round $i$ is noted $round[i].s\_box$
- ShiftRows is a rotation operation on each row of the State. The result of this operation at round $i$ is noted $round[i].s\_row$
- MixColumns is a linear matrix multiplication working on each column of the State. The result of this operation at round $i$ is noted $round[i].m\_col$
- AddRoundKey is a byte-wise XOR between the State and a value $k\_sch[i]$, computed from the key (according to a transform called *Key Expansion* which is not detailed in this paper). The result of this operation is the start of the next round noted $Round[i+1].start$.

The rounds are identical, except for the last one in which the Mixcolumns operation is skipped. At the end of the ciphering operations, the state is copied to the output as depicted in Figure 1.

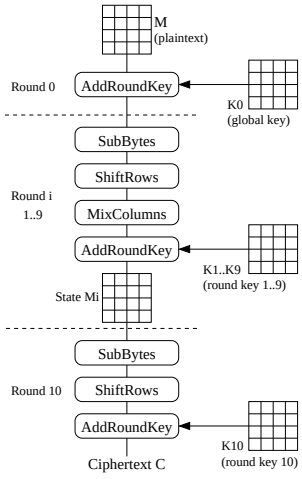


Fig. 1. $AES_{128}$ encryption algorithm.

## II. Proposed framework

### A. Attacker's environnement

*1) Observable data:* As explained above, physical attacks are based on experiments, i.e. the launches of cryptographic

operations on the target with or without modifying its behavior. During these experiments, the attacker is able to directly collect data such as digital values or analog ones. This set of values are further called *observable data*, or *observables*. Some observables, called *stimuli*, are used to launch the cryptographic operations and generally change at each experiment: plaintexts, whether there are perturbations or not, etc. Some observables also define the *experimental setup* and generally do not change during the experiments: kind of probes, gain and bandwidth of the amplifier, oscilloscope type, key value, etc. At last, some observables are the *reactions* or *responses* of the circuit according to the stimuli and the experimental setup. There are numerous reactions which can be recorded by the attacker:

- the EM, power traces or the signal provided by a micro-probe.
- the ciphertexts obtained without or with perturbations (*faulty ciphertexts*).
- the behavior (i.e. the normal or abnormal working) of the circuit in case of fault. This behavior may be determined, for example, by comparing the correct and faulty ciphering, by detecting the start of an alarm or not, the raise time of the alarm or by detecting a more or less premature stop in computation.
- the computation time. One way to measure this computation time consists in reducing the clock period step by step (on a specific clock cycle) and analyzing the circuit's behavior. The step in which the behavior of the circuit switches from normal to abnormal is considered as an indicator for the computation time. This particular step is called "fault intensity" in [LSG+10].

Note that we also consider that any combination of observables or any observable obtained by applying any signal processing technique (filtering, temporal to frequency transformation, etc.) is also considered as an observable.

*2) Hidden data:* At the opposite of observables, the execution of a cryptographic algorithm involves data that cannot be directly measured by the attacker. This piece of data is called *hidden data* or *internal data* in the following paragraphs.

*3) "Border" between observables and hidden data:* The *border* between observables and hidden data depends on the target and the attacker's means (i.e. laboratory equipment and threat scenario). In the following paragraphs, without loss of generality, the hidden data are the internal variables as defined in the standard (from $round[1].start$ included to $round[10].s\_row$ and every variable computed in the key schedule).

*4) Dimension of observables:* The measurements of observables may naturally be grouped in vectors or matrices of several dimensions. The power consumption, recorded with an oscilloscope, varies with the time. A power record (or *trace*) is thus a set of measurements which is grouped in a vector (dimension 1). An EM record, recorded with a EM probe moving in a plan above the chip, is naturally grouped in a 3-dimension matrix (time and X-Y location of the probe). For the sake of simplicity, we consider that a reaction is a scalar value (its dimension is equal to zero).

### B. Relationships between variables

As they are computed by or emanate from the same circuit, hidden data and observables are physically related to each others. These relationships can be described by a mathematical expression (for example between the plaintext, the key and the ciphertext in a nominal environment), some others cannot (for example between the plaintext, the key and the power consumption). But even if there is no mathematical expression for these relationships, the authors in [MR03], [MOS09] have introduced the notion of *leakage function* for side channel attacks. These functions formalize the relationship between some internal data (which are generally binary values) and some analog signals (such as power, EM, etc.). To extend this notion to fault attacks, we also consider "error leakage" functions. These functions formalize the modification by a fault of an internal data. As the internal data considered in physical attacks are generally for AES 8-bit values, each endomorphism of a set of 8 bits can be considered as a possible error function.

An *exploitable relationship* is a relationship between observables. This kind of relationship can be used to perform an attack by using a divide and conquer approach. Thus, such a relationship needs to involve only a partial key. Note that a relationship may be exploitable even without a mathematical expression. Examples of widely used exploitable relationships are reported in Table I. The relations only involves each byte of the state but for clarity are written, in this Table, for the whole state. In the following paragraphs, we note $p = R(F, k, O)$ the relation that links the reactions $p$ and a set of observables $O$ according to the value $k$ of the key and a set of leakage functions $F$.

### C. Relationship models

In the exploitable relationships described above, there is at least one unknown value: the partial key (or guess). So, the attacker has to create as many models as there distinct values of the guess key (i.e. 256 possibilities for an 8-bit guess key). For some relationships, other hidden data and one or several leakage functions also have to be guessed. It is impossible to test every possible leakage function because of their tremendous (even infinite in the case of side channel functions) number. The heuristics commonly used to choose one leakage function or a set of leakage functions are based on measurements or on a priori knowledge about the circuit and the measurement setup. Only the last case which concerns DPA-like, DFA and safe-error attacks, is described here. In this case, the models are built on the following small set of functions but other constructions are proposed for example in [DPRS11].

*1) Elementary functions:* A byte $B$ is a set of 8 bits noted $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$. The set with every bit set to one (resp. zero) is noted $\mathbb{1}$ (resp. $\mathbb{0}$). We match it with $B$ the binary value $B_b = b_7b_6b_5b_4b_3b_2b_1b_0$ but for the sake of simplicity, $B$ and $B_b$ will be used indifferently. The decimal value associated with $B$ is noted $B_d$ (for this conversion, $b_0$ is the less significant bit).

| Num | Observable | Internal | Relationship |
|---|---|---|---|
| $R_1$ | P=plaintext<br>O=micro-probing,EM, power, behavior | $k\_sch[0]$ | $O = f(round[1].start)$<br>with $round[1].start = SubBytes(AddRoundKey(P, k\_sch[0])$ |
| $R_2$ | P=plaintext<br>O=Power,EM | $k\_sch[0]$<br>INIT | $O = f(round[1].start), INIT)$<br>with $round[1].start = SubBytes(AddRoundKey(P, k\_sch[0])$ |
| $R_3$ | C=ciphertext<br>O=Micro-probing,EM, power, behavior | $k\_sch[10]$ | $O = f(round[10].start)$<br>with $round[10].start = SubBytes^{-1}(ShiftRow^{-1}(AddRoundKey(C, k\_sch[10])))$ |
| $R_4$ | C=ciphertext<br>O=faulted ciphertext | $k\_sch[10]$ | $O = AddRoundKey(ShiftRow(SubBytes(f(round[10].start))), k\_sch[10])$<br>with $round[10].start = SubBytes^{-1}(ShiftRow^{-1}(AddRoundKey(C, k\_sch[10])))$ |
| $R_5$ | C=ciphertext<br>O=faulted ciphertext or power | $k\_sch[10]$<br>$round[9].s\_row$ | $C' = h(AddRoundKey(round[10].s\_row', f(k\_sch[10])))$<br>with :<br>$round[10].s\_row' = ShiftRow(SubBytes(AddRoundKey(k\_sch[9]', round[9].s\_row)))$<br>$k\_sch[9]' = g(AddRoundKey(round[10].start, round[9].s\_row))$<br>$round[10].start = SubBytes^{-1}(ShiftRow^{-1}(AddRoundKey(C, k\_sch[10])))$ |

TABLE I
WIDELY USED RELATIONSHIPS

*a) Bit level functions:* The logical operations AND (symbol $\&$), OR (symbol $|$) and XOR (symbol $\oplus$) defined on a byte are considered bitwise.

*b) Weighted sum:* Let $B$ be a set of $N$ bits and $\Omega$ be a set of $N$ integer, real or binary numbers. The function $\sum_\Omega$ is the sum

$$\sum_\Omega(B) = b_N * \omega_N + b_{N-1} * \omega_{N-1} + \ldots + b_1 * \omega_1 + b_0 * \omega_0$$

*c) Select subset (or restriction) of bits:* Let $B$ (resp. $\Omega$) be a set of bits $b_i$ (resp. $omega_i$). The set of bits $b_i$ of $B$ such that $\omega_i$ of $\Omega$ is equal to "1", is noted $R_\Omega(B)$. Example: if $\Omega = \{0,0,1,0,0,1,0,0\}$, we have $\Omega_d = 36$ and the restriction of $B$ by $R_{36}$ is the set of bits $\{b_5, b_2\}$ of $B$. Note that if $B$ is constituted of $N$ values, there are $2^N - 1$ possible restrictions. A restriction is said *monobit* when only one bit of the byte is considered. So, the monobit restrictions are $R_1, R_2, R_4, \ldots, R_{128}$.

*d) Subset of bits equal to:* Let $\Omega$ and $B$ be two sets of $N$ bits. Let $M$ be the number of bits of the restriction of $R_\Omega(B)$ and $\alpha$ a set of $M$ bits. We note $R_\Omega(B) == \alpha$ the function that returns 1 when, for any bit $i$ of $R_\Omega(B)$ and $\alpha$, we have $R_\Omega(B)_i == \alpha_i$ and returns 0 otherwise.

*2) Side channel functions:* The simplest model proposed for power consumption consists in considering the value of a single bit. In this case, the leakage function is $R_i$ with $i \in \{1, 2, 4, \ldots, 128\}$. But more generally, side channel functions are chosen as a combination of selections and weighted sums. For example, the number of 1 values in a binary number $B$ is called *Hamming weight* and is noted $HW(B) = \sum_1(B)$. A variation consists in considering the number of transitions between two binary values. This function, called *Hamming distance*, is noted $HD(B, \Omega) = HW(B \oplus \Omega)$.

*3) Error functions:*

*a) Bit flip:* A fault may invert a set of bits. The function which models the bit flip is the bitwise XOR between $B$ and $\Omega$ with $\Omega_d \in [1, \ldots, 255]$. When all the bits of the bitflip $\Omega$ are equal to 1 (resp. 0), all the bits are inverted (resp. unchanged).

*b) Set and reset:* A fault may set a set of bits to specific values. The function which models the reset (resp. set) is the bitwise AND (resp. OR) between $B$ and $\Omega$ with $\Omega_d \in [1, \ldots, 255]$. A reset with $\Omega_d = 0$ returns a set of 0. A set with $\Omega_d = 255$ returns a set of 1.

*c) Behavior:* When a fault sets a set of bits $R_\Omega(B)$ of $B$ to some value $\alpha$, the circuit behavior is normal only when the perturbed result is expected to be equal to $\alpha$ and is abnormal otherwise. Such a behavior may be described by using the function $R_i(B) == \alpha$.

*d) Setup time violation:* When the faults are created by violating the setup time of latches, these faults may impact first the bytes which have the higher Hamming weight or may impact first only one bit. In the first case, chip's behavior may be described by a Hamming weight and in the second case by the function that selects one bit $R_i$ with $i \in \{1, 2, 4, \ldots, 128\}$.

### *D. Key retrieving algorithm*

In this part, we explain how to deal with exploitable relationships and measurements in order to discover a partial key. The first input of the algorithm 1 is a relationship between a set of observables and reactions. The models of this relationship is parameterized by the set of all the possible partial keys $K$ and a set of models $F$. This set of models formalizes the knowledge of the attacker about the circuit and the measurement setup. Two cases arise: in the first case, the attacker considers that one of these models is more probable than the others but he doesn't known which one is the good one (case "dist(F)=one"). In the second case, he considers that all the models are possible and are equiprobable (case "dist(F)=several"). The second input of the algorithm is the method to select the good or the bad key from a comparison between the prediction and the measurements. The different comparison methods are described in the algorithm 2. Roughly, when predictions and measurements are both discrete variables and when the "noise" in the measurements is considered negligible (it is typically the case for DFA), a sieve algorithm can be used. But when taking noise into account or when either predictions or measurements are continuous variables, some statistical distinguishers are preferred (a counting algorithm or a statistical similarity measurement). Mutual information (MIA) [GBT07], Pearson correlation [BCO03], distance of means (DoM) [KJJ99] or principal component (PCA)[SNG+10] are possible examples of such statistical similarity measurement tools.

Once these parameters are chosen, the measurements are processed in an incremental way, i.e the attacker tries to

recover this partial key with a reduced set of measurements. If no partial key can be distinguished, then the attack is performed with other measurements. He proceeds in this way until the partial key is recovered.

**Algorithm 1** Algorithm to distinguish the key (and the model) from the relationship $R$

**Require:** $p = R(K, F, O)$ : A relationship with:
$p$ : A reaction
$k \in K = \{k_0, \ldots, k_{2^\kappa - 1}\}$ : a set of $2^\kappa$ partial keys
$f \in F = \{f_1, \ldots, f_m\}$ : a set of $m$ leakage functions
$o \in O = \{o_1, \ldots, o_n\}$ : a set of $n$ observable
$dist(F) \in \{one, several\}$ the hypothesis on the distribution of leakage functions
$Max$ : Maximum number of measurements
$sim \in \{SIEVE, COUNT, CORR\}$ : an algorithm that computes a similarity
**Ensure:** $k_?$ : most "probable" key hypothesis
$o$ : a matrix (size $Max * n$) *The observables*
$p$ : a vector (size $Max$) *The reaction*
$Pred_p$ : a matrix (size $2^\kappa * m * Max$) *Prediction of the reaction p*
$c$ : a vector (size $2^\kappa$)
$Search \leftarrow TRUE$ ; $e \leftarrow 0$ ; $k_? \leftarrow NULL$
**while** $Search == TRUE$ and $e < Max$ **do**
  Perform the $e^{th}$ experiments
  Store the corresponding measurements in $o(e, 1:n)$
  Store the corresponding reaction in $p(e)$
  **for** $i = 0$ to $2^\kappa - 1$ **do**
    **for** $j = 1$ to $m$ **do**
      $Pred_p(i, j, e) = f_j(k_i, o(e, 1:n))$
    **end for**
  **end for**
  **for** $i = 0$ to $2^\kappa - 1$ **do**
    **if** dist(F)==one **then**
      $ct \leftarrow 0$
      **for** $j = 1$ to $m$ **do**
        $ct(j) \leftarrow sim(p(1:e), Pred_r(i, j, 1:e))$
      **end for**
      $c(i) \leftarrow MAX(ct(j))$
    **else**
      $c(i) \leftarrow sim(p(1:e), Pred_r(i, 1:m, 1:e))$
    **end if**
  **end for**
  **if** $\exists \alpha$ such that $\forall \beta$, $c(\alpha) >> c(\beta)$ **then**
    $k_? \leftarrow k_i$
    $Search \leftarrow FALSE$
  **else**
    $e \leftarrow e + 1$
  **end if**
**end while**

## III. Applications

In order to show that our formalism covers a wide class of attacks, the relationships (described in II-B) and the parameters for the algorithms (described in II-D) are reported in Table II for different attacks proposed in the literature. A simple but new attack is also proposed. It consists in probing the signal on one wire of a 8-bit data bus where the outputs of the Sboxes are supposed to pass through. Each leakage model corresponds to one bit of these outputs. If the attacker is able to temporally distinguish between the outputs of all the Sboxes, every method described above used to compute the similarity works. Otherwise, a method based on correlation is preferred. The attack scheme shows that the attacker may retrieve every bit of the key by probing only one single wire.

**Algorithm 2** Algorithm to compute the similarity

**Require:** : $Meas$ a vector of measurements (size $e$)
$Pred$ a matrix of measurements (size $m * e$)
$sim \in \{SIEVE, COUNT, CORR\}$ $corr \in DoM, CPA, MIA, PCA, \ldots$ statistical tool
**Ensure:** : $c$
**switch** $sim$
**case :**$SIEVE$
$c \leftarrow 1$
**for** $i = 1$ to $e$ **do**
  $l \leftarrow 0$
  **for** $j = 1$ to $m$ **do**
    $l \leftarrow l \quad | \quad Pred(j, i) == Meas(i)$
  **end for**
  $c \leftarrow c \& l$
**end for**
**end case**
**case :**$COUNT$
$c \leftarrow 0$
**for** $i = 1$ to $e$ **do**
  **for** $j = 1$ to $m$ **do**
    **if** $Pred(j, i) == Meas(i)$ **then**
      $c \leftarrow c + 1$
    **end if**
  **end for**
**end for**
**end case**
**case :**$CORR$
$index \leftarrow 1$
**for** $i = 1$ to $e$ **do**
  **for** $j = 1$ to $m$ **do**
    $ePred(index) \leftarrow Pred(j, i)$
    $eMeas(index) \leftarrow Meas(i)$
    $index \leftarrow index + 1$
  **end for**
  $c \leftarrow corr(ePred, eMeas)$
**end for**
**end case**
**end switch**

## Conclusion

This paper presents a formalism which is common for a wide class of physical attacks. It highlights that, despite their differences in term of experimental setup, all these attacks share the main underlying set of concepts and algorithms. This paper also shows that this set is relatively small. It enable to

| Attack | Relationship | Leakage functions and parameters and probablitity | Similarity and statistical tool |
|---|---|---|---|
| Probing | $R_1$<br>P=Plaintext<br>O=Probe value | $f(x) = R_\Omega(x)$ with<br>$\Omega \in \{1, 2, 4, \ldots, 128\}$<br>and $dist(F) = one$ | All<br>All |
| DPA<br>[KJJ99] | $R_3$<br>C=Ciphertext<br>O=Power | $f(x) = R_\Omega(x)$ with<br>$\Omega \in \{1, 2, 4, \ldots, 128\}$<br>and $dist(F) = one$ | CORR with<br>DoM |
| CPA<br>[BCO03] | $R_2$<br>P=Plaintext<br>O=Power | $f(x) = HW(x \oplus \Omega)$ with<br>$\Omega \in [\![1, 255]\!]$<br>and $dist(F) = one$ | CORR with<br>Pearson |
| MIA<br>[GBT07] | $R_1$<br>P=Plaintext<br>O=Power | $f(x) = HW(x)$ | CORR with<br>Mutual<br>Information |
| FSA<br>[LSG+10] | $R_3$<br>C=Ciphertext<br>O=Intensity | $f(x) = HW(x)$<br>or<br>$f(x) = R_\Omega(x)$ with<br>$\Omega \in \{1, 2, 4, \ldots, 128\}$<br>and $dist(F) = one$ | CORR with<br>Pearson |
| DBA<br>[RM07] | $R_1$<br>P=Plaintext<br>O=Behavior | $f(x) = (R_\Omega(x) == 0)$ with<br>$\Omega \in [\![1, 255]\!]$<br>and $dist(F) = one$ | CORR with<br>Pearson |
| DFA<br>[Gir05] | $R_4$<br>C=Ciphertext<br>O=Faulted<br>Ciphertext | $f(x) = x \oplus \Omega$ with<br>$\Omega \in [\![1, 255]\!]$<br>and $dist(F) = several$ | SIEVE |
| DFA<br>[RLK11] | $R_5$<br><br>C=Ciphertext<br>O=Faulted<br>Ciphertext | $h(x) = x$ and<br>$g(x, \Omega) = x \oplus \Omega$ with<br>$\Omega \in [\![1, 255]\!]$<br>and $dist(G) = one$<br>$f(y, \Gamma) = y \oplus \Gamma$ with<br>$\Gamma \in [\![1, 255]\!]$<br>and $dist(F) = one$ | COUNT |
| Combined<br>[RLK11] | $R_5$<br>C=Ciphertext<br>O=Power | $h(x) = HW(x)$<br>$f$ and $g$ as<br>defined above | CORR with<br>Pearson |

TABLE II
EXAMPLES OF APPLICATION OF THE FORMALISM

plan to define "new" attacks as new combinations of these concepts and algorithms. It will also enable to provide efficient and modular implementations of these attacks. Further works will consist in including in the proposed formalism template attacks, collision or algebraic attacks. Another future work will consist in integrating in our scheme the higher-order attacks.

## REFERENCES


[BCO03] Eric Brier, Christophe Clavier, and Francis Olivier. Optimal Statistical Power Analysis. *IACR Cryptology ePrint Archive*, 2003:152, 2003.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In B.S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[CT05] Hamid Choukri and Michael Tunstall. Round Reduction Using Faults. In *FDTC '05: Proceedings of the second Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 13–24, 2005.

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate Side Channel Attacks and Leakage Modeling. Cryptology ePrint Archive, Report 2011/302, 2011.

[GBT07] Benedikt Gierlichs, Lejla Batina, and Pim Tuyls. Mutual Information Analysis - A Universal Differential Side-Channel Attack. *IACR Cryptology ePrint Archive*, 2007:198, 2007.

[Gir05] Christophe Giraud. DFA on AES. In *Advanced Encryption Standard - AES*, volume 3373 of *Lecture Notes in Computer Science*, pages 27–41, 2005.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO*, pages 388–397, 1999.

[Koc96] Paul Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In *Advances in Cryptology - Crypto'96*, pages 104–113, New-York, 1996. Springer-Verlag.

[LSG+10] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault Sensitivity Analysis. In *CHES*, pages 320–334. 2010.

[MOS09] Stefan Mangard, Elisabeth Oswald, and François Xavier Standaert. One for All - All for One: Unifying Standard DPA Attacks. Cryptology ePrint Archive, Report 2009/449, 2009.

[MR03] Silvio Micali and Leonid Reyzin. Physically Observable Cryptography. Cryptology ePrint Archive, Report 2003/120, 2003.

[MSS06] Amir Moradi, Mohammad T. Manzuri Shalmani, and Mahmoud Salmasizadeh. A Generalized Method of Differential Fault Attack Against AES Cryptosystem. In *CHES*, pages 91–100, 2006.

[Muk09] Debdeep Mukhopadhyay. An Improved Fault Based Attack of the Advanced Encryption Standard. In Bart Preneel, editor, *Progress in Cryptology AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 421–434. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-02384-2_26.

[NIS01] NIST. Specification for the Advanced Encryption Standard. *FIPS PUB*, 197, November 2001.

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and Khazad. In C.D. Walter, editor, *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003.

[RLK11] T. Roche, V. Lomné, and K. Khalfallah. Combined Fault and Side-Channel Attack on Protected Implementations of AES. *Smart Card Research and Advanced Applications*, pages 65–83, 2011.

[RM07] Bruno Robisson and Pascal Manet. Differential Behavioral Analysis. In *CHES*, pages 413–426, 2007.

[SLIO12] K. Sakiyama, Y. Li, M. Iwamoto, and K. Ohta. Information-Theoretic Approach to Optimal Differential Fault Analysis. *Information Forensics and Security, IEEE Transactions on*, 7:109–120, 2012.

[SMP07] Elisabeth Oswald Stefan Mangard and Thomas Popp. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer Verlag, 2007.

[SNG+10] Youssef Souissi, Maxime Nassar, Sylvain Guilley, Jean-Luc Danger, and Florent Flament. First Principal Components Analysis: A New Side Channel Distinguisher. In Kyung Hyune Rhee and DaeHun Nyang, editors, *ICISC*, volume 6829 of *Lecture Notes in Computer Science*, pages 407–419. Springer, 2010.

[TM09] M. Tunstall and D. Mukhopadhyay. Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault. Cryptology ePrint Archive, Report 2009/575, 2009.

[YJ00] Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Transactions on Computers*, 49(9):967–970, 2000.